CSC301 AI Assignment Report

Team Name	301_programmers	Sub-Team Name	AI Chat Titles
Sub-Team Members	Maya Edri, Olivia Wong, Maleeha Rahman		
Video Presentation Link	https://youtu.be/N-uB9G_rINY		
PR Link	https://github.com/csc301-2025-y/project-16-rivvi/pull/48		
README Link	https://github.com/csc301-2025-y/project-16-rivvi/blob/feature/ai-chat-titles/README.md		

Part 1 – Planning:

1. Identifying the problem/improvement
Our application lets employees ask policy-related questions via AI. Currently, chat logs are named by date, making it hard to locate past conversations. To improve UX, we use AI to generate content-based titles from the first message, making chat history easier to navigate.

2. Criteria, Research, Selection
The criteria for our AI feature was based on the following key points:

- **Clarity of user value and Improved UX**
  - Helps users quickly recognize and locate specific conversations in a long history
  - Reduces cognitive load by replacing vague labels like "New chat" or "Untitled" with descriptive summaries
  - Titles can be tailored to reflect the actual intent or topic of the conversation
- **Feasibility with Current AI Capabilities**
  - Can be implemented using existing language models that excel at summarization and title generation
  - Only requires a single input (the first user message), making it simple to implement compared to more complex multi-turn summarization
- **Low Data Requirements**
  - Doesn't require long context or full conversation history
    - can generate a meaningful title just from one message
  - Makes it suitable even for systems where full chat history isn't stored for privacy reasons
- **Efficiency and Automation**
  - Can run in the background instantly after the first message is sent
  - Requires no additional input or action from the user, ensuring a seamless user experience

Potential AI solutions, models, frameworks, and libraries suitable for our goal:
Models:

- OpenAI GPT-4 / GPT-3.5: excel at summarization, paraphrasing, and title generation; can analyze the first user message and produce concise, descriptive titles based on the topic, tone, or intent
- Gemini API: also supports summarization and natural language tasks

Libraries / Frameworks:
- **LangChain**: Useful for chaining prompts, formatting input/output, and managing memory in more complex AI flows
- **Hugging Face:** gives access to various summarization and sequence-to-sequence models that can be adapted for title generation

Integration tools:
- **FastAPI**: Already being used for our backend, and it's well-suited for exposing a lightweight AI microservice that receives a user's message and returns a generated title; supports async execution and integrates well with LangChain and OpenAI libraries
- **Firebase**: Can be used to store generated titles and link them to unique chat sessions or users
- **Next.js / React**: For rendering the AI-generated titles in the chat UI (e.g., in a sidebar or search view)

We chose **Gemini API** since it was already integrated into our backend, minimizing setup and avoiding extra API keys, rate limits, or vendor wrappers. Its summarization and input handling made it ideal for generating concise titles quickly.

We used **LangChain** to structure prompts, manage memory, and format output. It integrates well with Gemini, enabling quick prototyping with minimal overhead. Although we considered Hugging Face, it would have required extra work to host models or use their APIs, making LangChain the more practical choice.

For infrastructure, we reused tools from our existing stack: **FastAPI** to expose a lightweight AI service, **Firebase** to store titles and sync them with users and chats, and **Next.js** with **React** to display the titles in the UI. This made integration smooth and efficient.

We use a pre-trained language model (e.g., GPT) to generate titles, so no custom training or datasets are needed. The model is already suited for summarizing short inputs. In the future, we may collect anonymized message-title pairs to improve results (prompt tuning).

- Data Preprocessing: minimal preprocessing is needed since we use only the first user message. We may remove extra whitespace or greetings (e.g., "Hi") if they reduce quality.
- Bias Mitigation: risk is low, as the model generates short, descriptive titles. Still, we may add filters to prevent offensive or misleading output.
- Ethical Consideration:
    - Transparency - users should be aware that titles are generated automatically.
    - Privacy: titles should never be stored/exposed beyond the user's own interface.
    - Non-assumptive language: titles will reflect the user's message content and do not infer intent inaccurately.

This feature improves the usability by automatically generating a short, descriptive title from the user's first message. As conversations increase, it becomes harder to identify past chats. Summarizing each topic helps users locate specific discussions without opening them individually. This supports our goal of making workplace policies more accessible via a better UX.

Part 2 – Implementation:

Model Selection

- We selected **Google's Gemini LLM** to generate chat titles. Since it was already integrated in our backend, setup was minimal and consistent with our stack. Gemini's strong summarization and NLP capabilities make it ideal for concise titles, offering an efficient balance of speed, cost and output quality.

Data Pipeline Setup

- After a chat ends, the full transcript is stored in Firebase
- A backend process sends this transcript to the Gemini LLM through a secure API route.
- The LLM responds with a concise title, which is then saved to the database and displayed in the chat history UI.

API Integration

- We built a custom API route in our Next.js backend to securely communicate with the Gemini LLM
- The route includes prompt engineering logic to instruct the model to generate short, descriptive titles.

Development Steps

We first identified the need for chat title generation, designed prompts using CURL for testing, then implemented a backend API route to send input to Gemini. We updated the frontend to display titles, and validated the feature with end-to-end and backend tests.

Obstacles

- Chat History Not Updating with New Title: the title didn't appear in chat history due to a mismatch in the *userId* used in backend logic. After debugging, we corrected the user ID handling to ensure titles were properly stored in Firebase and displayed in the UI.
- Premature Title Generation: titles were being generated before the first message was sent, resulting in empty or irrelevant titles. We fixed this by updating the logic to trigger title generation only after the first message was saved.

Part 3 – Impact Analysis:

Performance comparison:

- Before AI: chat logs were labeled with date-based names (e.g., Chat - 07/10/2025), making it difficult to navigate long chat histories.
- After AI: titles are now content-based (e.g., "Overtime Pay Roles"), improving searchability and recall of chats.

Testing methodology: We manually tested end-to-end by creating chats with different opening messages, verifying title update after the first message, and confirming persistence in Firebase. Backend Testing: We used CURL to test the API route connected to the LLM, ensuring it returned appropriate titles and that responses were handled correctly before frontend integration.

Trade-offs: This feature adds minimal server and token load—only one short message is processed—while significantly improving UX by making chat history easier to navigate.